開発環境説明書

はじめに

表記について

本取扱説明書では、安全にお使いいただくために、守っていただきたい事項に次のマークを表示しております。

	人体に危険を及ぼしたり、装置に大きなダメージを与えたりする可能性があることを示しています。必ずお守りください。
	機能停止を招いたり、各種データを消してしまったりする可能性があることを示しています。十分に注意してください。
	関連する情報を記載しています。参考にお読みください。

商標について

「Rooster」は、サン電子株式会社の登録商標および商標登録出願中です。
Oracle と Java は、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。
その他、本取扱説明書に記載されている会社名、製品名は、各社の商標または登録商標です。
本文中の各社の商標または登録商標には、TM、®マークは表示しておりません。

GPL／LGPLライセンスについて

本製品は、GPL version2.0／LGPL version2.0 の適用ソフトウェアを使用しております。オープンソースとしての性格上著作権による保証はなされておりませんが、本製品につきましては保証書、および取扱説明書記載の条件により当社による保証がなされています。GPL／LGPL のライセンスにつきましては、以下の URL をご覧ください。

- http://www.gnu.org/licenses/gpl-2.0.html
- http://www.gnu.org/licenses/lgpl-2.0.html

変更済み GPL 対象モジュール、その配布方法につきましては、サン電子（株）サポートセンターにご連絡ください。なお、配布時発生する費用はお客様のご負担となります。

▶ 本取扱説明書の画面イメージは開発中のものです。
実際の画面とは多少異なる場合があります。

安全上のご注意（必ずお守りください）

ここに記載している注意事項は、安全に関わる重要な内容ですので、必ず守ってください。本取扱説明書では、安全上の注意事項を「警告」と「注意」に区分しています。

警告　この表示を無視して、間違った取り扱いをした場合、人が死亡または重傷を負う可能性が想定される内容を示しています。

この表示を無視して、間違った取り扱いをした場合、人が損害を負う可能性が想定される内容、および物的損害のみの発生が想定される内容を示しています。物的損害とは、家屋、家財および家畜、ペットに関する拡大損害を示しています。

禁止行為（してはいけないこと）を示しています。

強制行為（必ずしなければいけないこと）を示しています。

なお、注意、禁止に記載した事項でも、状況によっては重大な結果に結びつく場合があります。いずれも重要な内容を記載していますので、必ず守ってください。

警 告

分解禁止

本製品を分解したり、改造したりしないでください。

➔ 感電、火災、故障の原因になります。

禁止

近くに雷が発生したときには AC アダプタまたは電源ケーブルを本体から抜いてご使用をお控えください。

➔ 落雷が火災、感電、故障の原因となるときがあります。

禁止

本製品に水などの液体をかけたり、異物を入れたりしないでください。

➔ 感電や火災の原因になります。

万一、本製品に液体がかかったり、異物が入ったりした場合は、AC アダプタまたは電源ケーブルを本体から抜いて、点検修理を依頼してください。

強制

製品から煙、異臭、異常音が発生した場合は、AC アダプタまたは電源ケーブルを本体から抜き、本製品を接続している機器からケーブルを取り外してください。また、点検修理を依頼してください。

➔ 火災の原因になります。

禁止

電源ケーブルを傷つけないでください。

➔ 感電、火災の原因になります。

強制

AC アダプタは、AC100V コンセントに接続してください。また、本製品を設置、移動する時は、電源プラグを抜いてください。

➔ 故障、火災の原因になります。

禁止

梱包のポリ袋などは、小さいお子様の手の届く所に置かないでください。

➔ 小さいお子様がかぶったり、飲みこんだりすると、呼吸を妨げる危険があります。

強制

電源プラグは確実に根元まで差し込んでください。また、電源プラグとコンセントの間のほこりは、定期的（半年に一回程度）に取り除いてください。

➔ 電源プラグの間にほこりが付着し、電源が短絡して発煙、発火、火災の恐れがあります。

注 意

禁止

この取扱説明書に記載されている周囲環境条件以外では、使用、保管しないでください。

➔ 本製品の故障や破損などによって、発煙、発火、感電の原因になります。下記の環境には、特にご注意ください。

- 室内または製品周囲の温度や湿度が極端に高い、または低い場所
- 結露がある場所
- 急激な温度変化が起きる場所
- ほこりが多い場所
- 静電気が発生しやすい場所
- 腐食性のガスが発生する場所
- 水などがかかりやすい場所
- 振動や衝撃が加わるような不安定な場所
- 油煙が当たる場所
- 直射日光が当たる場所
- 製品周囲に発熱する器具や燃えやすい物がある場所
- 周囲に置いてある物との間に適切な空間がない場所

禁止

専用の AC アダプタまたは規格に合った電源以外を使用しないでください。

➔ 他の電源を使用すると、故障、火災の原因になります。

強制

30cm 以上の高さから落とした場合は、使用を中止し、点検、修理を依頼してください。

➔ そのまま使用すると、重大な事故になる可能性があります。

ご使用にあたってのお願い

- 本製品周辺で静電気的障害を発生させないでください。
  ➔ 本製品は、静電気に敏感な部品を使用しています。特に、コネクタの接点、ポート、その他の部品に、素手で触れないでください。部品が静電破壊するおそれがあります。
- 本製品はていねいに取り扱ってください。
  ➔ 本製品に強いショックを与えると破損の原因になります。
- 本製品のお手入れは、電源を切った状態で行ってください。
  ➔ 誤動作や故障の原因になります。
- 本製品のお手入れには、揮発性の有機溶剤、薬品、化学ぞうきんなどを使用せず、乾いた柔らかい布で拭いてください。汚れがひどい場合は、柔らかい布に台所中性洗剤をしみこませて固く絞ってから拭き、最後に乾いた柔らかい布で仕上げてください。
  ➔ 揮発性の有機溶剤、薬品、化学ぞうきんなどを使用すると、変質、変色、場合によっては破損の原因になります。

地球環境保全のため、次のことにご協力ください。

- 本製品および付属品は、不燃物として処分してください。
- 廃棄方法は、地方自治体などで決められた分別収集方法に従ってください。
- 一般ごみとして、家庭で焼却処分しないでください。
  ダイオキシンや塩化水素ガスなどが発生し、環境や人体に影響を与えます。

ご注意

- 本製品の仕様は国内向けになっておりますので、海外ではご利用になれません。

  These products are designed for use in Japan only and cannot be used in any other countries.
- 本製品は、パソコンなどの OA 機器に使用することを目的に設計、製造されています。医療機器や幹線通信機器、電算機システムなどの、きわめて高い安全性や信頼性が要求される用途には使用しないでください。
- 一般の電話機やテレビ・ラジオなどをお使いになっている近くで使用すると、影響を与える場合がありますので、なるべく離れた場所でご使用ください。
- 強い磁界の中や腐食性のガスの中で使用したり保管したりしないでください。
  故障の原因となります。

本装置に電源を供給して使用した場合、下記の事項を注意することを推奨いたします。

- 高精度な制御や微弱な信号を取り扱う電子機器の近くでは、本装置の電源を切れる構造とすることをお奨めします。
  電子機器が誤作動するなど影響を与える可能性があります。
  【ご注意いただきたい電子機器の例】
  補聴器、植込み型心臓ペースメーカおよび植込み型除細動器、その他医用電気機器、火災報知機、自動ドア、その他の自動制御機器など

  ※参考：「医用電気機器への電波の影響を防止するための携帯電話端末等の使用に関する指針」（電波環境協議会[平成 9 年 4 月]）
- 取扱説明書について、次の点にご注意ください。
  1. 本製品は各通信事業者の USB 型データ通信端末を利用して無線によるデータ通信を行う事が出来る装置です。本製品およびモバイル通信端末等の不具合、誤動作又は停電、回線障害、その他の外部要因によって通信障害が発生したために生じた損害等については、当社としては責任を負いかねますので、あらかじめご了承ください。
  2. 本取扱説明書の内容の一部または全部を、無断で転載することを禁止します。
  3. 本取扱説明書の内容に関しては、将来予告なしに変更される場合があります。
  4. 本取扱説明書の内容につきましては、万全を期して作成致しましたが、万一ご不審な点や、ご不明な点、誤り、記載漏れ、乱丁、落丁、その他お気づきの点等ございましたら、当社までご連絡ください。
  5. 適用した結果の影響につきましては、3 項にかかわらず責任を負いかねますので、ご了承ください。
  6. 本取扱説明書で指示されている内容につきましては、必ず従ってください。本取扱説明書に記載されている内容を無視した行為や誤った操作によって生じた障害や損害につきましては、保証期間内であっても責任を負いかねますので、ご了承ください。

目次

1章　開発環境の概要

この章では、Rooster GX の開発環境の概要、および開発環境を使用するにあたって必要なネットワーク構成とソフトウェア構成について説明します。

1-1 開発環境について

Rooster GX の開発環境を利用すると、Rooster GX の機能を最大限に活かした独自のアプリケーションを開発することができます。
Rooster GX は、以下の開発環境を提供しています。

- VMware Player を使用したクロス開発環境
- NFS マウント方式のファイルイメージを利用したネイティブ開発環境

それぞれの開発環境の Linux ディストリビューションには Debian 6.0 squeeze を採用しているため、開発に必要なパッケージを容易にインストールすることが可能です。

➲ ディストリビューションの詳細については、『3 章　ディストリビューションの概要』で説明しています。

1-2 開発環境に含まれるソフトウェア

開発環境に添付しているメディアには、次のものが含まれています。

firmwareディレクトリ

Rooster GX のファームウェアが保存されています。また、MD5 チェックサム値が保存されているファイルも格納されています。

▶ MD5 チェックサム値は、`md5sum` コマンドで作成しています。

nfsディレクトリ

ネイティブ開発環境である NFS イメージが保存されています。

▶ 添付されている VMware イメージには、NFS イメージが含まれています。

sdkディレクトリ

Rooster GX ファームウェアを作成するための SDK が保存されています。

▶ 添付されている VMware イメージには、SDK が含まれています。

sourcesディレクトリ

Rooster GX で採用しているブートローダー（U-Boot）とカーネル（Linux-2.6.31）のソースファイルが保存されています。

toolsディレクトリ

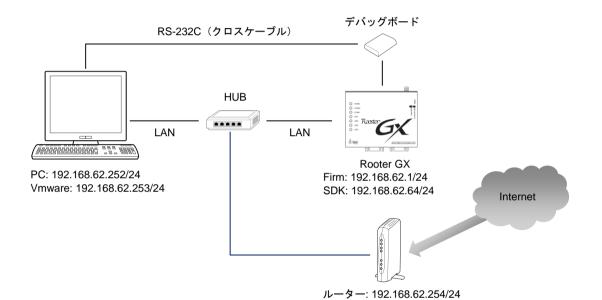
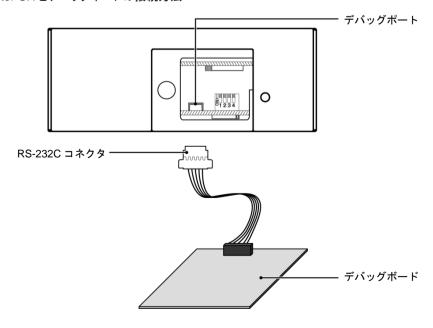
クロス開発環境である VMware イメージが保存されています。

1-3 開発環境の接続構成

開発環境を利用するためには、PC（Windows パソコン）と Rooster GX を以下の図のように接続してください。

社内 LAN には接続せず、必ず独立したネットワークとして構成するようにしてください。

ルーターは、クロス開発環境やネイティブ開発環境のパッケージ追加／更新時などに Internet へ接続するために使用します。Internet を利用しない場合は、ルーターに接続する必要はありません。

開発環境の接続構成

Rooster GX とデバッグボードの接続方法

2章　開発環境の起動

この章では、クロス開発環境とネイティブ開発環境を起動するための設定方法および起動手順について説明します。

2-1 クロス開発環境の起動

ここでは、クロス開発環境を起動し、ネイティブ開発環境の起動に必要なサービス環境設定について説明します。

2-1-1　クロス開発環境の設定

VMware Player を使用して、クロス開発環境の VMware イメージファイルを起動します。

VMware Player Version 5.0.2 build-1031769 で動作確認しています。

1. クロス開発環境の VMware イメージファイル「Debian 6 Rooster-GX.zip」を、「マイドキュメント¥Virtual Machine」などのフォルダに解凍します。
   ▶「マイドキュメント¥Virtual Machine」以外のフォルダでも問題ありません。

2. VMware Player の設定をして、VMware イメージファイルを起動します。

3. 以下のアカウントでログインします。

ID	パスワード
root	root
user	user

ネイティブ開発環境の起動に必要な設定をするには、root 権限が必要となります。

2-1-2　サービス環境設定

ここでは、主要なサービスの構成について説明します。
ネットワーク環境にあわせてサービスを設定してください。

SSHサービス環境

SSH サービスは、起動時に自動的に開始されるようになっています。

DHCPサービス環境

DHCP サービスは、起動時に自動的に開始されないように設定されています。
ネットワーク環境にあわせて設定を変更してください。『1-3　開発環境の接続構成』で示した構成で利用する場合は、Rooster GX の MAC アドレスの設定のみを変更するだけで、DHCP サービスを開始することができます。

1. エディタで/etc/dhcp/dhcpd.conf ファイルを開き、ネットワークの設定を変更します。

   /etc/dhcp/dhcpd.conf の設定例

```
dns-update-style none;
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 192.168.62.0 netmask 255.255.255.0 {
  range 192.168.62.32 192.168.62.63;
  option routers 192.168.62.254;
  option subnetmask 255.255.255.0;
  option broadcast-address 192.168.62.255;
  host rooster {
    hardware ethernet 00:80:F3:6F:xx:xx;      ←Rooster GX の MAC アドレスを指定
    fixed-address 192.168.62.64;
    filename "uImage";
    option host-name "roosterGX";
    option root-path "/srv/nfs/roosterGX";
    next-server 192.168.62.253;
  }
}
```

2. DHCP サーバを起動します。

```
# /etc/init.d/isc-dhcp-server start ↵
```

   DHCP サーバを停止するには、次のように実行します。

```
# /etc/init.d/isc-dhcp-server stop ↵
```

3. 起動時に自動的に開始するように設定するには、次のように実行します。

```
# update-rc.d isc-dhcp-server defaults ↵
```

   自動的に開始しないように設定するには、次のように実行します。

```
# update-rc.d isc-dhcp-server remove ↵
```

TFTPサービス環境

TFTP サービスは、接続要求が発生した場合に、自動的に起動される設定になっています。
ネイティブ開発環境の起動時に、Linux カーネルイメージを配信します。

- カーネルイメージは、「/srv/tftp/uImage」に保存してください。
- カーネルイメージは、VMware イメージに含まれています。

NFSサービス環境

NFS サービスは、ネイティブ開発環境に必要です。
ネットワーク環境にあわせて設定する必要があります。ただし、『1-3　開発環境の接続構成』で示した構成で利用する場合は、設定する必要はありません。

1. エディタで`/etc/exports` ファイルを開き、設定を変更します。

   `/etc/exports` の設定例

   ```
   /srv/nfs/roosterGX 192.168.62.0/24(insecure,rw,sync,no_root_squash)
   ```

2. 設定を変更した場合、以下のコマンドを実行して、設定内容を反映してください。

   ```
   # exportfs -ra ↵
   ```

2-2 ネイティブ開発環境の起動

ここでは、ネイティブ開発環境を起動する手順について説明します。

1. PC と Rooster GX を接続します。

   ➲ 接続の構成と注意事項については、『1-3 開発環境の接続構成』を参照してください。

2. VMware Player でクロス開発環境の VMware イメージファイルを起動し、DHCP／TFTP／NFS サービスを起動します。

   ➲ 詳細については、『2-1-1 クロス開発環境の設定』を参照してください。

3. Rooster GX の DIP スイッチをブートローダモード（U-Boot）に設定し、電源を供給します。

   ➲ DIP スイッチについては、『付録 A ハードウェアに関する情報』を参照してください。

4. コンソールに「Enter password - autoboot in 60 sec...」と出力されてから、60 秒以内に「roostergx」と入力します。

5. 「SUNCORP>>」というコマンドプロンプトが表示されたら、以下のように DHCP を実行します。

```
Enter password - autoboot in 60 sec...
SUNCORP>> dhcp ↵
BOOTP broadcast 1
DHCP client bound to address 192.168.62.64
```

6. DHCP で正常に IP アドレスを取得できたら、run bootcmd_nfs コマンドを実行します。

```
SUNCORP>> run bootcmd_nfs ↵
Using egiga0 device
TFTP from server 192.168.62.253; our IP address is 192.168.62.64
Filename 'uImage'.
Load address: 0x2000000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #############################
done
Bytes transferred = 3149384 (300e48 hex)
## Booting image at 02000000 ...
   Image Name:     Linux-2.6.31.β
   Created:        2013-02-04 7:11:56 UTC
   Image Type:     ARM Linux Kernel (uncompressed)
   Data Size:      3149320 Bytes = 3 MB
   Load Address:   00008000
   Entry Point:    00008000
   Verifying Checksum ... OK
OK

Starting kernel ...
```

```
Uncompressing
Linux......................................................................
...........................................................................
............. done, booting the kernel.
Linux version 2.6.31.8 (user@debian) (gcc version 4.4.5 (Debian 4.4.5-8) ) #1 Mon Feb
4
16:11:40 JST 2013
CPU: Feroceon 88FR131 [56251311] revision 1 (ARMv5TE), cr=00053977
CPU: VIVIT data cache, VIVIT instruction cache
Machine: Feroceon-KW

<省略>
Debian GNU/Linux 6.0 rooster ttyS0

rooster login:
```

7. 「rooster login:」プロンプトが表示されたら、以下のアカウントでログインします。

ID	パスワード
root	root
user	user

8. 通常、ルートディレクトリは read only に設定されているため、書き込むことができません。
ルートディレクトリを書き込み可能に設定するためには、以下のコマンドを実行する必要があります。

```
# mount -o remount,rw / ↵
```

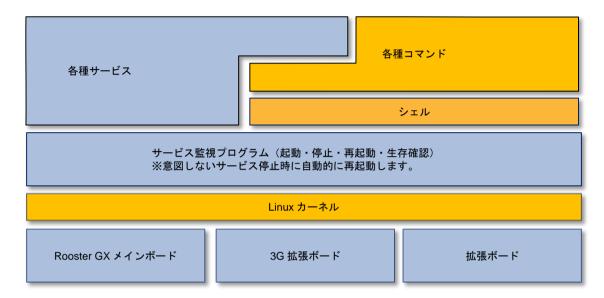
read only に戻すには、次のコマンドを実行します。

```
# mount -o remount,ro / ↵
```

3章 ディストリビューションの概要

Rooster GX の開発環境で使用している Linux ディストリビューションは、Debian 6.0 squeeze で提供されているソースファイルをベースに、サン電子が独自の修正を加えたオリジナルディストリビューションです。この章では、ディストリビューションやファイルシステムの構成、利用できるオープンソースなど、サン電子オリジナルディストリビューションの概要について説明します。

3-1 ディストリビューション構成

サン電子のオリジナルディストリビューションは、次のように構成されています。

3-2 ファイルシステム構成

ここでは、ファイルシステムの構成および主要なファイルについて説明します。

3-2-1 ファイルマウント構成

サン電子オリジナルのディストリビューションは、基本的には Debian 6.0 squeeze と同じ構成ですが、組み込みシステム向けに、マウントポイントなどが修正されています。
ディストリビューションのファイルシステム構成を以下に示します。

ファイルシステムとマウントポイントの一覧

デバイス	デバイスパス	マウントパス	ファイルシステムフォーマット
MTD4	/dev/mtdblock4	/	JFFS2(ro)
MTD5	/dev/mtdblock5	/var/log	JFFS2(rw)
none	proc	/proc	proc
none	sysfs	/sys	sysfs
none	tmpfs	/mnt/union	tmpfs
none	unionfs	/etc	/mnt/union/etc
none	unionfs	/var	/mnt/union/var
none	unionfs	/tmp	/mnt/union/tmp
none	unionfs	/media	/mnt/union/media
none	tmpfs	/dev	tmpfs(udev)

3-2-2 Version情報ファイル

ディストリビューションの Version 情報は、`/etc/version` ファイルに保存されています。

/etc/version の例

```
Rooster GX Version xx.xx.xx build xx
```

▶ xx には可変長で数字が入ります。

3-2-3 設定ファイル

Rooster GX の設定パラメータは、`/etc/config.xml` ファイルに XML 形式で保存されています。
`config save` コマンドを実行すると、Flash メモリに保存されます。

➲ コマンドの詳細については、『Rooster GX 取扱説明書』の『コマンドリファレンス編』を参照してください。

3-2-4 SYSLOGファイル

SYSLOG は、ログファイルの `/var/log/messages` に保存されます。指定サイズを超えた場合、ログファイルはローテートされます。

3-3 オープンソース一覧

サン電子オリジナルのディストリビューションは、以下のオープンソースを使用して作成されています。

オープンソース一覧（標準）

パッケージ名	バージョン	備考
attr	2.4.44	
bind9	9.7.3	
bsdmainutils	8.0.13	
busybox	1.17.1	http://www.busybox.net/
bzip2	1.0.5	
curl	7.21.0	
cyrus-sasl2	2.1.23	
daemontools	0.76	http://cr.yp.to/daemontools.html
db	4.7	
db	4.8	
debianutils	3.4	
e2fsprogs	1.41.12	
eglibc	2.11.3	
ethtool	2.6.34	
findutils	4.4.2	
gcc	4.4.5	http://gcc.gnu.org/
gdbm	1.8.3	
geoip	1.4.7	
gmp	4.3.2	
gnutls26	2.8.6	
iproute	20100519	http://www.policyrouting.org/
ipsec-tools	0.7.3	
iptables	1.4.8	http://www.netfilter.org/index.html
keyutils	1.4	
krb5	1.8.3	
libbsd	0.2.0	
libcap2	2.19	
libedit	2.11-20080614	
libgcrypt11	1.4.5	
libgpg-error	1.6	
libidn	1.15	
libnfnetlink	1.0.0	
libpcap	1.1.1	
libselinux	2.0.96	
libssh2	1.2.6	
libtasn1	3.2.7	
libusb	0.1.12	

パッケージ名	バージョン	備考
libxml2	2.7.8	http://xmlsoft.org/
lsof	4.8.1	
lzo2	2.03	
minicom	2.4	
mtd-utils	20090606	
ncurses	5.7	
netbase	4.45	
ntp	4.2.6.p2	http://www.ntp.org/index.html
openldap	2.4.23	
openssh	5.5p1	http://www.openssh.com/
openssl	0.9.8o	
openswan	2.6.28	
pam	1.1.1	
pdnsd	1.2.7	
perl	5.10.1	
ppp	2.4.5	http://ppp.samba.org/
quagga	0.99.20.1	http://www.nongnu.org/quagga/
readline5	5.2	
readline6	6.1	
rp-pppoe	3.8	
strace	4.5.20	http://sourceforge.net/projects/strace/
tcpdump	4.1.1	http://www.tcpdump.org/
tcp-wrappers	7.6.q	
udev	164	http://www.kernel.org/
util-linux	2.17.2	
xfsprogs	3.1.4	http://oss.sgi.com/projects/xfs/
xz_utils	5.0.0	
zlib	1.2.3.4	

オープンソース一覧(無線 LAN 対応)

パッケージ名	バージョン	備考
dbus	1.2.24	
expat	2.0.1	
glib2.0	2.24.2	
libcap-ng	0.6.4	
libnl	1.1	
libpcre3	8.02	
pcsc-lite	1.5.5	
wireless-tools	30pre9	

3-4 クロスコンパイル環境

VMware Player を使用したクロス開発環境には、次のプログラムやスクリプトが用意されています。

3-4-1　クロスコンパイル用プログラム

クロスコンパイル用に、以下のプログラムがインストールされています。

```
arm-linux-gnueabi-addr2line
arm-linux-gnueabi-ar
arm-linux-gnueabi-as
arm-linux-gnueabi-c++filt
arm-linux-gnueabi-cpp
arm-linux-gnueabi-cpp-4.4
arm-linux-gnueabi-g++
arm-linux-gnueabi-g++4.4
arm-linux-gnueabi-gcc
arm-linux-gnueabi-gcc-4.4
arm-linux-gnueabi-gcov
arm-linux-gnueabi-gcov-4.4
arm-linux-gnueabi-gprof
arm-linux-gnueabi-ld
arm-linux-gnueabi-nm
arm-linux-gnueabi-objcopy
arm-linux-gnueabi-objdump
arm-linux-gnueabi-ranlib
arm-linux-gnueabi-readelf
arm-linux-gnueabi-size
arm-linux-gnueabi-strings
arm-linux-gnueabi-strip
```

3-4-2　ファームウェア作成環境

ファームウェアを作成するためのビルドスクリプトが用意されています。

➲ 詳細については、『5-3　ファームウェアの作成』を参照してください。

4章　サービスの管理

この章では、サービスを監視するための仕組みやサービスの登録と制御方法などについて説明します。

4-1 サービス監視プログラム

Rooster GX の開発環境では、サービス監視プログラムのデーモンツール（daemontools）によって、対象のサービスプロセスを監視することができます。
サービス監視プログラムが run スクリプトを実行し、run スクリプトから exec コマンドで監視対象のプロセスをフォアグラウンドで実行することで、監視対象のプロセスを監視することが可能となります。

down ファイルが存在する場合、run スクリプトは実行されません。

!

デーモンとしてバックグラウンドで実行されている場合、正常に監視できません。

正常・異常の関係なしに、監視しているサービスプロセスが終了した場合、サービス監視プログラムが再度 run スクリプトを実行します。

起動時と同様に、down ファイルが存在する場合、run スクリプトは実行されません。

respawn 開始

サービス監視プログラム

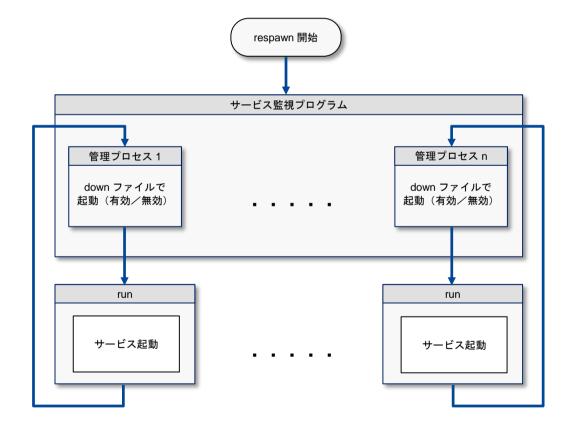
管理プロセス 1
down ファイルで
起動（有効／無効）

・・・・・

管理プロセス n
down ファイルで
起動（有効／無効）

run
サービス起動

・・・・・

run
サービス起動

4-2 監視するサービスの管理

サービスプロセスをデーモンツールの監視対象として登録し、必要に応じてサービスを実行／停止する手順について説明します。

4-2-1 サービスの登録

サービスプロセスを監視対象として登録するには、次の手順で実行します。

1. /var/service/***ServiceName*** ディレクトリを作成します。

   ▶ ***ServiceName*** は、サービスごとの名前に置き換えてください。

2. 手順 1 で作成したディレクトリに run スクリプトを作成し、ファイルのパーミッションを 700 に設定します。

```
# chmod 700 /var/service/ServiceName/run ↵
```

3. Rooster GX の起動時にサービスを自動的に起動したくない場合は、手順 1 で作成したディレクトリに空の down ファイルを作成します。

```
# touch /var/service/ServiceName/down ↵
```

4. /etc/service に、手順 1 で作成したディレクトリのシンボリックリンクを作成します。

```
# ln -s /var/service/ServiceName /etc/service/ ↵
```

4-2-2 サービスの実行

監視対象のサービスプロセスを実行する方法には、次の 2 種類があります。

- ファームウェアを作成して起動

  ➲ ファームウェアの作成方法については、『5-3 ファームウェアの作成』を参照してください。

- 制御コマンドを実行

```
# svc -u /etc/service/ServiceName ↵
```

4-2-3 サービスの停止

以下の手順で監視対象のサービスプロセスを停止することができます。

- down ファイルを作成

```
# touch /etc/service/ServiceName/down ↵
```

- 制御コマンドを実行してサービスを停止

```
# svc -tx /etc/service/ServiceName ↵
```

4-2-4 監視対象サービスの削除

監視中のサービスプロセスを監視対象から外すには、次の手順で実行します。

1. /etc/service/***ServiceName*** のシンボリックリンクを削除します。

2. 制御コマンドを実行して、サービスを停止します。

```
# svc -tx /var/service/ServiceName ↵
```

5章 アプリケーションの開発

この章では、アプリケーションのパッケージを作成して、ファームウェアを作成／更新する方法について説明します。

5-1 アプリケーションのコンパイル方法

アプリケーションのコンパイルは、Rooster GX と同じハードウェア環境でアプリケーションを開発可能な、NFS マウント方式のファイルイメージを利用したネイティブ開発環境で実行することをお勧めします。

➔ ネイティブ開発環境の起動方法については、『2-2 ネイティブ開発環境の起動』を参照してください。

5-2 パッケージの作成

パッケージを作成するために必要なディレクトリやファイルの配置場所や設定内容、プログラムやライブラリの設定、パッケージの圧縮方法の手順について説明します。

以下の手順では、/home/user/project をプロジェクトディレクトリ、/home/user/project/rootfs をファームウェアディレクトリと呼びます。

5-2-1 基本ディレクトリの作成

最初に、プロジェクトの基本となるディレクトリを作成します。

1. プロジェクトの基本ディレクトリを作成します。

```
$ cd /home/user ↵
$ mkdir -p project/rootfs ↵
```

5-2-2 設定ファイルの保存と復旧

Rooster GX では、設定ファイルを Flash メモリに保存することができます。

1. config.list ファイルを編集し、設定リストにファイルを追加します。

```
$ cd /home/user/project/rootfs ↵
$ mkdir -p etc/default ↵
$ vi etc/default/config.list ↵
```

etc/default/config.list の設定例

```
/etc/config.list
/etc/config.xml
/etc/fstab
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/prg1.conf      ←追加
```

追加行以外の行（赤字の行）はすべて必要なファイルです。必ず記述するようにしてください。

2. Rooster GX にすで保存されているデータがある場合、ファームウェアを更新した後に、設定の初期化処理を実行する必要があります。

   INIT Push Switch を押したままの状態で Rooster GX を起動するか、Rooster GX のデバックコンソールで以下のコマンドを実行することで、設定を初期化できます。

```
# config initialize ↵
```

設定の初期化が必要なのは、config.list ファイルを修正した場合のみです。

3. Rooster GX で以下のコマンドを実行して、設定を Flash メモリに保存します。

```
# config save ↵
```

4. 手動で設定を復旧するには、以下のコマンドを実行します。

Rooster GX の起動時に、保存されているファイルは自動的に復旧されます。

```
# config load ↵
```

5-2-3　プログラムとライブラリの設定

パッケージに必要なプログラムとライブラリの設定をします。

1. ファームウェアディレクトリは Rooster GX のルートディレクトリ（/）に割り当てられていますので、ファームウェアディレクトリに必要なファイルを保存します。

この作業は root 権限で実行してください。

たとえば、prg1 プログラムをファームウェアの/usr/sbin に置きたい場合は、ファームウェアディレクトリ/home/user/project/rootfs に/usr/sbin ディレクトリを作成し、prg1 をコピーします。

また、prg1 が参照するライブラリ libprg1 も同様にライブラリディレクトリに保存します。

```
$ su - ↵              ←root 権限に移行
# cd /home/user/project/rootfs ↵
# mkdir -p usr/sbin ↵
# cp -a Path/prg1 usr/sbin ↵
# mkdir -p usr/lib ↵
# cp -a Path/libprg1.so usr/lib ↵
```

5-2-4 監視対象のサービスとして登録

プログラムを監視対象のサービスとして登録します。

➲ サービスの監視については、『4 章　サービスの管理』も参照してください。

1. ファームウェアディレクトリの var/service/project を作成し、/etc/service にリンクを設定します。

   また、var/service/project/run ファイルと var/service/project/down ファイルを作成します。

- この作業は root 権限で実行してください。
- サービス（デーモン）プログラムは、フォアグラウンド処理で実行する必要があります。fork しないようにしてください。バックグラウンドで実行した場合、監視することができませんので、注意が必要です。

```
$ su - ↵              ←root 権限に移行
# cd /home/user/project/rootfs ↵
# mkdir -p var/service/project ↵
# touch var/service/project/down ↵
# ln -s /var/service/project etc/service/ ↵          ←/の入力に注意
# vi var/service/project/run ↵
```

var/service/project/run の設定例

```
#!/bin/bash

##
# environment setting
SERVICE_NAME=sample
LOG="/usr/bin/logger -t ${SERVICE_NAME}"
CHKRUN=/var/run/syslogd.pid
CHKCNT=100

##
# load library
. /usr/lib/sun/svcfunc

##
# starting udevd wait
udevd_childs_to_finish

##
# starting syslog wait
check_pid ${CHKRUN} ${CHKCNT}

${LOG} -p user.info "Starting the ${SERVICE_NAME} service: ${SERVICE_NAME}."
exec /usr/sbin/${SERVICE_NAME} > /dev/null 2>&1
```

5-2-5 パッケージの圧縮

ファームウェア作成用のパッケージファイルを作成します。

1. tar と bzip2 でファームウェアディレクトリを圧縮して、パッケージを作成します。

```
# cd /home/user/project ↵
# tar jcf prg1.tar.bz2 rootfs ↵
```

ファームウェア作成用のパッケージファイル prg1.tar.bz2 が作成されます。

5-3 ファームウェアの作成

作成したパッケージからファームウェアを作成する手順について説明します。

1. RoosterGX-SDK ディレクトリに移動します。

```
$ cd /home/user/RoosterGX-SDK ↵
```

2. 『5-2　パッケージの作成』で作成した prg1.tar.bz2 パッケージを、パッケージディレクトリ pkgs にコピーします。

```
$ cp prg1.tar.bz2 /home/user/RosoterGX-SDK/pkgs/ ↵
```

3. build.conf ファイルにパッケージを登録します。

build.conf ファイルの PKG 環境変数にパッケージ名を追加することで、ファームウェアにパッケージを追加することができます。

```
$ sudo vi build.conf ↵        ← vim エディタを起動
[sudo] password for user:     ← 「user」と入力
```

build.conf ファイルの設定例

```
##
# Set build versions
VENDOR=suncorp                ← ファームの拡張名を記入
EXTRAVERSION=-v1.0.0          ← 拡張ファームのバージョンを記入

##
# Package options

##
# Packages not supported
# PKG="pkg1 pkg2 pkg2"
PKG="パッケージ名を登録"
```

パッケージ名に拡張子は含みません。パッケージファイル名が ***prg1***.tar.bz2 の場合、パッケージ名は ***prg1*** となります。

4. ファームウェアファイルを作成します。

   以下のコマンドを実行することで、imgs ディレクトリにファームウェアが作成されます。

```
$ sudo ./build ↵
[sudo] password for user:   ←「user」と入力
Rootfs...
  install roosterGX package
  install prg1 package
  success !
Firmware...
  Note: Use space JFFS2 (27180008/94371840)[28%]
  making firmware image
  making check sum
  success !
--------------------------------------------------------------------------------
Rooster GX version 1.2.0 build 9 (suncorp-v1.0.0) (Mon, 25 Feb 2013 11:48:10 +0900)
--------------------------------------------------------------------------------
build succeeds!
```

   build.conf を参照せずにパッケージを追加したい場合は、以下のように「-p パッケージ名」というオプションを付けて実行します。

```
$ sudo ./build -p prg1 ↵
<省略>
```

   カーネルを変更したい場合は、「-k カーネルファイル名」オプションを付けて実行します。

```
$ sudo ./build -k /home/user/linux-2.6.31.8/arch/arm/boot/uImage ↵
<省略>
```

5-4 ファームウェアの更新

作成したファームウェアを更新する方法には、DIP スイッチによる更新方法と、firm コマンドによる更新方法の 2 種類があります。
ここでは、それぞれの更新方法の手順について説明します。

5-4-1 DIPスイッチによる更新

DIP スイッチを「ファームウェア更新モード」に設定して起動することによって、サーバから自動的にファームウェアファイルを取得し、ファームウェアを更新することができます。

1. ファームウェアファイルをクロス開発環境イメージの/srv/tftp ディレクトリにコピーします。

2. クロス開発環境イメージの DHCP サービス設定を変更します。

/etc/dhcp/dhcpd.conf の設定例

```
dns-update-style none;
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;
log-facility local7;

subnet 192.168.62.0 netmask 255.255.255.0 {
  range 192.168.62.32 192.168.62.63;
  option routers 192.168.62.254;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.62.255;
  host rooster {
    hardware ethernet 00:80:F3:6F:xx:xx;     # Rooster GX の MAC アドレスを指定
    fixed-address 192.168.62.64;
    filename "roosterGX-vX_X_X-bX.frm";      # ファームウェアイル名を指定
    option host-name "roosterGX";
    option root-path "/srv/nfs/roosterGX";
    next-server 192.168.62.253;
  }
}
```

3. クロス開発環境イメージの DHCP サービスを再起動します。

```
# /etc/init.d/isc-dhcp-server restart ↵
```

4. Rooster GX の DIP スイッチを「ファームウェア更新モード」に変更して、電源を ON にします。

5-4-2 firmコマンドによる更新

Linux が起動している状態でファームウェアを更新する場合は、`firm` コマンドを実行します。

1. DIP スイッチを「通常起動モード」で起動します。

2. WinSCP などのツールを使用して、`/tmp` ディレクトリにファームウェアファイルをコピーします。

3. デバッグコンソールまたは SSH などを利用して、以下のコマンドを実行します。

```
# firm -n /tmp/roosterGX-vX_X_X-bX.frm -m update ↵
```

5-5 Javaの起動方法

作成した Java アプリを起動する方法の手順について説明します。

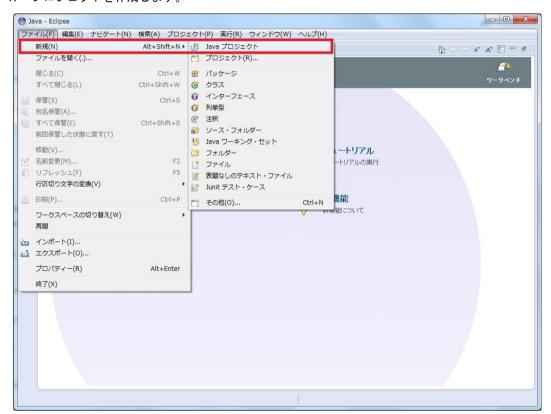
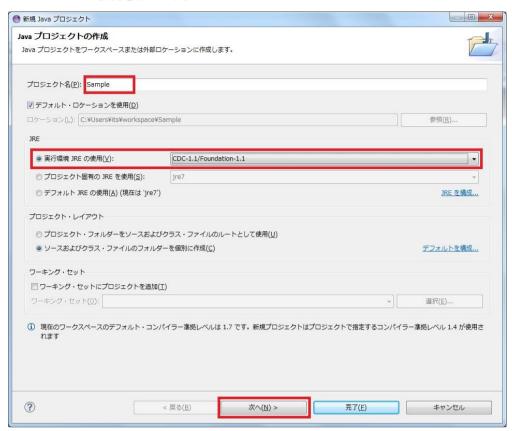
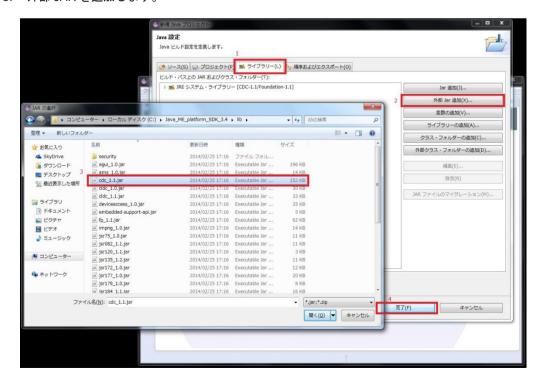
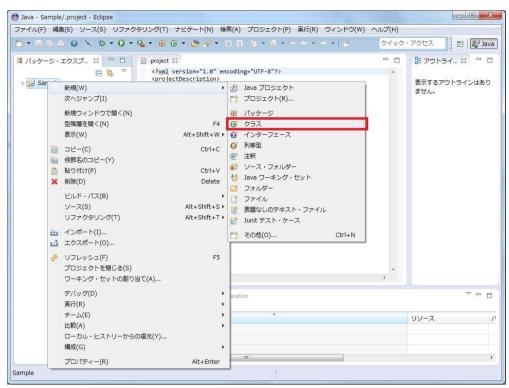
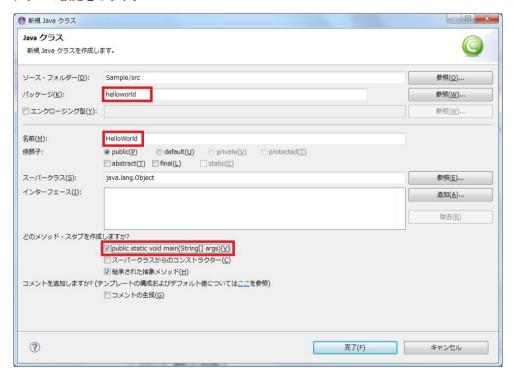
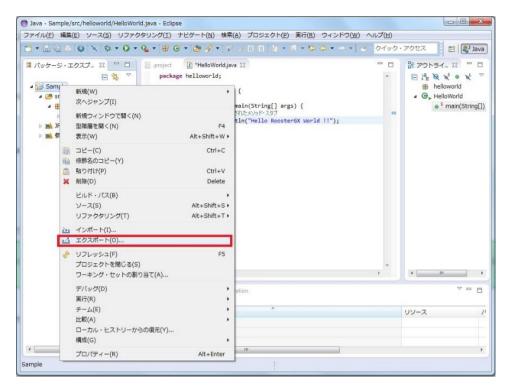
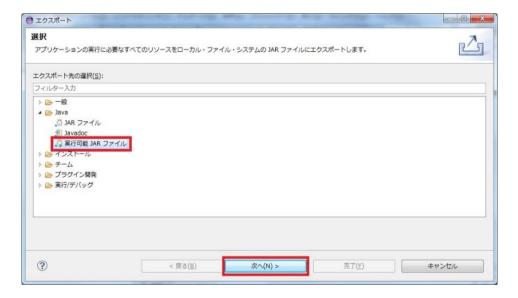
5-5-1 開発環境

下記の開発環境を使用し、動作することを確認しています。

1. Java SE Development Kit 7（http://www.oracle.com/technetwork/java/javase/downloads/index.html）
2. Java ME SDK 3.4（http://www.oracle.com/technetwork/java/javame/javamobile/download/sdk/index.html）
3. Kepler Service Release 1 (Ver 4.3.1)（http://www.eclipse.org/downloads/）

5-5-2 作成方法

下記の手順で作成した jar ファイルを RoosterGX で実行することができます。

1. プロジェクトを作成します。

2. プロジェクトの設定を行います。

3. 外部 JAR を追加します。

4. クラスを作成します。

5. クラスの設定をします。

6.　作成したプログラムを RoosterGX で実行するため、エクスポートします。

7.　エクスポートを実行可能 JAR で行います。

8. ファイル仕様を設定します。

9. エクスポートした JAR ファイルを RoosterGX にコピーし、下記コマンドにて実行することができます。

実行例

```
# cvm -Xjit:aotFile=/tmp/cvm.aot /tmp/sample.jar↵
```

```
192.168.62.1:22 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) 漢字コード(K) ヘルプ(H)
Last login: Tue Feb 25 18:07:01 2014 from 192.168.62.2


BusyBox v1.17.1 (Debian 1:1.17.1-8) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ #
~ #
~ #
~ #
~ # cvm -Xjit:aotFile=/tmp/cvm.aot -jar /tmp/sample.jar
Hello RoosterGX World !!
~ #
```

5-5-3　RoosterGXのJava環境確認方法

下記のコマンドを実行することで動作環境を確認できます。

例

```
# cvm -Xjit:aotFile=/tmp/cvm.aot -version↵
```

付録

付録A　ハードウェアに関する情報

Rooster GX のハードウェア仕様を以下に示します。

RS-232Cポート

RS-232C のデバイス名

デバイス名	ポート名	備考
/dev/ttyS0	デバッグポート	デバッグポート接続用
/dev/ttyS2	PORT0	外部ポート
/dev/ttyS3	PORT1	外部ポート
/dev/ttyS4	ユーザ非公開	アクセスしないでください。

デバッグポート

デバッグポート信号一覧

ピン No.	信号名	入出力	内容
2	RxD(RD)	IN	受信データ
3	TxD(SD)	OUT	送信データ

デバッグポート設定

機能	設定内容
通信速度（bps）	115200
データ長（ビット）	8
パリティ	No Parity
ストップビット	1
フロー制御	None

外部ポート

外部ポート信号一覧

ピン No.	信号名	入出力	内容
1	DCD(CD)	IN	キャリア検出
2	RxD(RD)	IN	受信データ
3	TxD(SD)	OUT	送信データ
4	DTR(ER)	OUT	データ端末レディ
6	DSR(DR)	IN	データセットレディ
7	RTS(RS)	OUT	送信リクエスト
8	CTS(CS)	IN	送信可
9	RI(CI)	IN	被呼表示

外部ポート設定一覧

機能	設定内容
通信速度（bps）	300／600／1200／2400／4800／9600／19200／38400／57600／115200
データ長（ビット）	7／8
パリティ	Odd／Even／Mark／Space／No Parity
ストップビット	1／2
フロー制御	RTS/CTS（ハードウェア）、XON/XOFF（ソフトウェア）、None

LED

LED 状態説明

LED 状態	記号	補足
消灯	●	
点灯	○	
点滅	☼	点灯と消灯を繰り返す状態です。
点灯消灯	◑	点滅より速く点灯と消灯を繰り返す状態です。

LED 点灯･点滅パターン一覧

状態	POWER	ETHER 1	ETHER 2	LED 1	2	3	4	補足
通電状態	○							通電時は POWER が点灯します。
LAN1 Link 状態		○						LAN が Link 状態時に点灯します。
LAN1 送受信時		☼						データ送受信時に点滅します。
LAN2 Link 状態			○					LAN が Link 状態時に点灯します。
LAN2 送受信時			☼					データ送受信時に点滅します。
ブートローダ起動時				○	○	○	○	起動時に約 500ms 間点灯します。
カーネル起動中				○1	○2	○3	○4	カーネル起動の進行状態を表します。 LED1、LED2、LED3、LED4 の順番で点灯します。
カーネル停止処理中				●4	●3	●2	●1	カーネル停止処理の状況を表します。 開始時に全点灯し、LED4、LED3、LED2、LED1 の順番で消灯します。
設定初期化中				◑	◑	◑	◑	設定初期化中は、LED1～4 が同時に点灯消灯を繰り返します。約 500ms 間隔で点灯･消灯を繰り返します。
ファームウェア更新中				◑	◑	◑	◑	ファームウェア更新中は、 LED1～4 が同時に点灯消灯を繰り返します。約 500ms 間隔で点灯・消灯を繰り返します。
異常発生時				○	○	○	○	すべての LED が点灯します。
カーネル停止時				●	●	●	●	すべての LED が消灯します。
PORT0 送受信時				◑				カーネル起動後に PORT0 が送受信した場合に点灯します。約 100ms 後に消灯します。
PORT1 送受信時					◑			カーネル起動後に PORT1 が送受信した場合に点灯します。約 100ms 後に消灯します。
通信モジュール回線状態						○ ↓ ●		回線接続時に点灯し、回線切断時に消灯します。
通信モジュールアンテナ状態							◑	点灯、点滅パターンについては、以下の表『通信モジュールアンテナ状態』を参照してください。

通信モジュールアンテナ状態

状態	LED 点灯パターン	補足
圏外等通信不可時		LED は消灯します。
弱い（1 本）		
やや弱い（2 本）		
普通		

DIPスイッチ

DIP スイッチパターン表

DIP スイッチ				動作モード
1	2	3	4	
ー	ON	ON	ON	ファームウェア更新モード
ー	ON	ON	OFF	ブートローダモード
ー	ON	OFF	ON	microSD カード起動モード
ー	ON	OFF	OFF	ユーザ設定起動モード
ー	OFF	ON	ON	DSR 検出無効モード
ー	OFF	ON	OFF	
ー	OFF	OFF	ON	
ー	OFF	OFF	OFF	通常起動モード

※ DIP スイッチ変更時は、電源 OFF 状態を 1 分以上継続後に、電源を ON する必要があります。

シリアル番号の確認方法

RoosterGX のシリアル番号は本体の裏面の製造番号の下 5 桁です。

付録B カーネルコンパイル手順

カーネルのコンパイルは、クロス開発環境で実行することができます。
ここでは、カーネルコンパイルの基本的な手順を示します。

カーネルのコンパイルは、カーネルオプションを変更したい場合にのみ実行してください。

```
$ make CROSS_COMPILE=arm-linux-gnueabi- mrproper ↵
$ cp arch/arm/configs/rooster_gx_defconfig ./.config ↵
$ make CROSS_COMPILE=arm-linux-gnueabi- oldconfig ↵
$ make CROSS_COMPILE=arm-linux-gnueabi- menuconfig ↵    ←カーネルオプションを変更する場合
$ make CROSS_COMPILE=arm-linux-gnueabi- uImage modules ↵
$ cp arch/arm/boot/uImage /srv/tftp/uImage ↵          ←NFS 起動用カーネルイメージを更新する
```

付録C　アプリケーション開発に役立つコマンド

Rooster GX の開発では、以下のコマンドを利用して、機器の状態表示や制御用にアプリケーションを開発することができます。

アプリケーション作成コマンド

コマンド	概要
scpu	CPU 間通信コマンド
led	LED 制御コマンド
firm	ファームウェア更新コマンド

CPU間通信コマンド

scpu コマンドでは、「CPU 間データ通信サービス IF」を利用して、各種設定をしたり、情報を取得したりできます。

scpu の基本書式

```
scpu [-h] { version | config | heartbeat | status | reset | rtc | usb-hub | gpio | mcpu }
```

scpu のパラメータ

パラメータ	概要	詳細
version	サブ CPU のファームウェアバージョン番号を標準出力に出力します。 出力例：Version 1.0.0	
config	サブ CPU の情報を取得／設定します。	『config』
heartbeat	Heart Beat をサブ CPU に送信します。	
status	状態を取得して、指定した情報をコンソールに出力します。	『status』
reset	機器を再起動します。	
rtc	RTC の情報を取得／設定します。	『rtc』
usb-hub	USB-HUB チップのレジスタ値を取得／設定します。	『usb-hub』
gpio	サブ CPU 配下の GPIO 端子の設定値を取得／設定します。	『gpio』
mcpu	メイン CPU の動作状態を取得／設定します。	『mcpu』
-h	標準出力に Usage を出力します。	

正常終了時は、戻り値として 0 が返されます。異常終了時は 0 以外が返されます。

config

サブ CPU の設定を取得・設定します。

config のパラメータ

```
scpu config { temp | volt | port | mpf | event }
```

パラメータ	概要	
temp { **ha** | **la** | **hw** | **lw** }	温度監視設定をします。 以下のパラメータを設定します。温度の入力範囲は-128～128（℃）です。	
	ha ***High Low***	温度（高）異常の閾値を設定します。
	la ***High Low***	温度（低）異常の閾値を設定します。
	hw ***High Low***	温度（高）警告の閾値を設定します。
	lw ***High Low***	温度（低）警告の閾値を設定します。
volt { **ha** | **la** | **hw** | **lw** }	電圧監視設定をします。 以下のパラメータを設定します。電圧の入力範囲は 0～23（V）です。	
	ha ***High Low***	電圧（高）異常の閾値を設定します。
	la ***High Low***	電圧（低）異常の閾値を設定します。
	hw ***High Low***	電圧（高）警告の閾値を設定します。
	lw ***High Low***	電圧（低）警告の閾値を設定します。
port ***u1 u2 um***	DSR による省電力制御設定をします。 以下のパラメータを設定します。	
	u1	port0 を設定します。 0： 何もしない 1： 省電力に移行のみ 2： 省電力から復帰のみ 3： 両方対応（復帰・移行）
	u2	port1 を設定します。 0： 何もしない 1： 省電力に移行のみ 2： 省電力から復帰のみ 3： 両方対応（復帰・移行）
	um	port0／1 の省電力制御の優先設定をします。 0： 通常優先※1 1： port0 のみ 2： port1 のみ 3： 省電力優先※2 ※1 port0／1 どちらかの DSR が ON の場合は省電力から復帰します。また、両方の DSR が OFF になった場合に省電力に移行します。 ※2 port0／1 両方の DSR が ON の場合は省電力から復帰します。また、どちらかの DSR が OFF になった場合に省電力に移行します。 ※3 u1／u2 の設定が優先されます。
mpf ***Count Time***	瞬停検出精度を設定します。 以下のパラメータを設定します。	
	Count	0～255 の数値を設定します。
	Time	0～255（ms）の数値を設定します。

パラメータ	概要
event { **save** | **back** | **reset** | **log** }	定期実行を設定します。 以下のパラメータを設定します。 save ***E Week Month Day Hour Minute***　省電力移行イベントを設定します。 指定できるパラメータを以下に示します。 E：　スケジュールの有効・無効 1：有効 0：無効 Week：週設定（0〜6、9） 0：日曜 1：月曜 2：火曜 3：水曜 4：木曜 5：金曜 6：土曜 9：すべての曜日 Month：月設定（1〜12、99） 99：すべての月 Day：　日設定（1〜31、99） 99：すべての日 Hour：　時設定（0〜23、99） 99：すべての時 Minute：分設定（0〜59、99） 99：すべての分 back ***E Week Month Day Hour Minute***　省電力復帰イベントを設定します。 ➲ 指定できるパラメータについては、上記の save のパラメータを参照してください。 reset ***E Week Month Day Hour Minute***　定期的に機器を再起動します。 ➲ 指定できるパラメータについては、上記の save のパラメータを参照してください。 log ***E Week Month Day Hour Minute***　定期的に状態ログを保存します。 ➲ 指定できるパラメータについては、上記の save のパラメータを参照してください。

status

状態を取得して、指定した情報をコンソールに出力します。

status のパラメータ

```
scpu status [ dsw | psw | temp | volt ]
```

パラメータ	概要	
dsw	取得した DSW を、標準出力に 16 進数文字列で出力します。	
psw { **init** | **stop** | **current** }	取得したビット情報を、標準出力に 16 進数文字列で出力します。 以下のパラメータを指定します。	
	init	設定初期化フラグを出力します。
	stop	機器停止フラグを出力します。
	current	現在値を出力します。
temp	取得した温度情報を摂氏温度に変換し、標準出力に小数点 2 桁で出力します。	
volt	取得した電圧情報を電圧に変換し、標準出力に小数点 2 桁で出力します。	

reset

機器の再起動を実行します。

reset のパラメータ

```
scpu reset type sec [ reason ]
```

パラメータ	概要	
dsw	取得した DSW を、標準出力に 16 進数文字列で出力します。	
type	再起動種別を以下から指定します。	
	hw	システムフルリセットを実行します。
	cpu	CPU リセットを実行します。
sec	コマンドを実行してから再起動するまでの時間を秒で指定します。	
reason	再起動理由を以下から指定します。 省略された場合、manual が選択された状態となります。	
	manual	手動でのリセットを表します。
	firmware	ファームウェア更新による再起動を表します。

rtc

UTC（Real Time Clock）の情報を取得／設定します。パラメータなしで実行した場合は、RTC（Real Time Clock）を取得してコンソールに出力します。パラメータを指定した場合は、RTC を設定します。

rtc のパラメータ

```
scpu rtc [utc] [ YYMMDDhhmmss ]
```

パラメータ	概要
utc	LOCAL TIME から UTC 時間に変換して時刻を設定します。
YYMMDDhhmmss	YY:年（下二桁） MM:月 DD:日 hh:時 mm:分 ss:秒

usb-hub

USB-HUB チップのレジスタ値を取得します。

usb-hub のパラメータ

```
scpu usb-hub Addr [ Value ]
```

パラメータ	概要
Addr	レジスタアドレスを指定します。
Value	設定値を指定します。省略された場合、レジスタアドレスの値を取得し、標準出力に 16 進数文字列で出力します。

gpio

サブ CPU 配下の GPIO 端子の設定値を取得／設定します。

gpio のパラメータ

```
scpu gpio { config | data }
```

パラメータ	概要
config [***Value***]	GPIO 設定値を設定します。***Value*** が省略された場合、現在値を標準出力に 16 進数で出力します。
data ***Num*** [***Value***]	***Num*** 番の GPIO 設定値を設定します。***Value*** が省略された場合、現在値を標準出力に 16 進数で出力します。

mcpu

メイン CPU の動作状態を取得／設定します。
Value が省略された場合、現在値を標準出力に 16 進数で出力します。

mcpu のパラメータ

```
scpu mcpu [ Value ]
```

LED制御コマンド

led コマンドでは、LED を制御することができます。

led の基本書式

```
led [ -m { set | clear | blink | shot | ant0 | ant1 | ant2 | ant3 | port0 | port1 | modem }
| -p ] [ -d Device ] [ -n Num ] [ -h ]
```

led のパラメータ

パラメータ	概要	詳細
-m { **set** | **clear** | **blink** | **shot** | **ant0** | **ant1** | **ant2** | **ant3** | **port0** | **port1** | **modem** }	LED の制御モードを指定します。	➲『LED の制御モード』
-p	LED モードを表示します。	
-d ***Device***	LED デバイスを指定します（指定不要です）。	
-n ***Num***	設定・取得する LED 番号を ***Num*** に指定します。 1　LED1 2　LED2 3　LED3 4　LED4 all　すべての LED	
-h	標準出力に Usage を出力します。	

正常終了時は、戻り値として 0 が返されます。異常終了時は 0 以外が返されます。

LEDの制御モード

LED の制御モードとして、以下のパラメータを指定します。

-m のパラメータ

```
led -m { set | clear | blink | shot | ant0 | ant1 | ant2 | ant3 | port0 | port1 | modem }
```

パラメータ	概要
set	LED を点灯させせます。
clear	LED を消灯させせます。
blink	LED を点滅させせます。
shot	LED が set 状態で shot が指定された場合、100ms 間消灯します。 clear 状態で shot が指定された場合、100ms 間点灯します。 ▶ LED が set または clear 以外の状態で shot が指定された場合は、無視されます。
ant0～ant3	アンテナレベルを指定します。 ant0　圏外 ant1　弱い ant2　やや弱い ant3　普通

パラメータ	概要	
port0〜port1	RS-232C のポート番号を指定します。	
	port0	port0 の送受信発生時に点滅します。
	port1	port1 の送受信発生時に点滅します。
modem	3G モジュールの送受信発生時に点滅します。	

ファームウェア更新コマンド

firm コマンドでは、ファームウェアを更新することができます。

※　サブ CPU のファームウェアは更新できません。将来対応予定です。

firm の基本書式

```
firm -n FirmFile -m { update | check | version } [ -w RebootWaitTime ] [ -v Level ] [-h]
```

firm のパラメータ

パラメータ	概要	詳細
-n ***FirmFile***	***FirmFile*** にファームウェアファイルを指定します。	
-m { update | check | version }	動作モードを指定します。	➲『動作モード』
-w ***RebootWaitTime***	正常にファームウェアが更新された場合、正常終了時から指定した秒数が経過した後に、再起動します。 このパラメータが省略された場合、正常終了後ただちに再起動します。	
-v ***Level***	詳細情報を出力します。 設定する数値が大きいほど、詳細な情報を出力します。 設定範囲： 1～7 の数値	
-h	標準出力に Usage を出力します。	

正常終了時は、戻り値として 0 が返されます。異常終了時は 0 以外が返されます。

動作モード

動作モードとして、以下のパラメータを指定します。

-m のパラメータ

```
firm -m { update | check | version }
```

パラメータ	概要
update	ファームウェアを更新します。
check	ファームウェアに問題がないかどうかを確認します。 ファームウェアは更新しません。
version	ファームウェアファイル情報を出力します。 ファームウェアは更新しません。

付録D　機器の状態情報

Rooster GX の開発では、下記のファイルを参照することにより、機器の状態を確認することが出来ます。

瞬停による再起動確認

/proc/driver/mcpu/power ファイルには、瞬停による再起動かそうでないかを識別するための情報が保存されています。

値	内容
0	通常起動
1	瞬停による起動

瞬停情報取得例

```
$ cat /proc/driver/mcpu/power
0
```

Watchdogタイマーによる再起動確認

/proc/driver/mcpu/watchdog ファイルには、メイン CPU（ハードウェア/ソフトウェア）の Watchdog タイマーによる再起動かそうでないかを識別するための情報が保存されています。

値	内容
0	通常起動
1	Watchdog タイマーによる再起動

Watchdog タイマー情報取得例

```
$ cat /proc/driver/mcpu/watchdog
0
```